Generic Model Interpretations: POSIX.1 and SQL

D. Elliott Bell*

Mitretek Systems
7525 Colshire Drive
McLean VA 22102

Abstract

An improvement to the traditional process of model interpretation is described. The improvement applies to trusted systems that conform to industry standards that are conducive to generic model interpretation. Generic model interpretation results for POSIX.1 and SQL are presented.

KEYWORDS: Security model, model interpretation, POSIX, SQL, TCSEC

Introduction

The modeling requirements for Division B of the *Trusted Computer System Evaluation Criteria* (*TCSEC*) [TCSEC85] derive from the very earliest experiences in conceiving and producing trusted computer systems. The Anderson report [AND72] called for a "conceptual design, that is a mathematical model" of a secure computer system as part of the plan to realize the "Reference Monitor Concept" in an implementation. Efforts during the 1970's and early 1980's (see for example [WALT74], [BLP73], [LPB73], [BELL73], [BLP75], [GOME82], [GOME84]) included conceptual design tools in the form of mathematical models for use in assessing the (defined) security of systems and products of interest.

Work before (and system and product evaluation since) the publication of the *TCSEC* has been very consistent in terms of the elements and the correspondences that are required in a model interpretation of a specific implementation. In *TCSEC* terms, what is required is a Formal Security Policy (FSP), a Formal Security Policy Model (FSPM), a Descriptive Top-Level Specification (DTLS), and connections between them.

The required connection between the model and the policy is that it be a model *of* the policy in question so that modeling results will allow policy-relevant statements to be made about the system at issue. The model itself must also be shown to be sound and free from defect.

There must also be a (descriptive top-level) specification of the system. This abstraction of the system at its interface must be shown to *correspond* to the model. The notion is that the model results — usually of the form "this transition preserves the important notions of 'security' " — are inherited by the system at the level of the specification if the individual system calls (or gates or entry points) exhibit the same behavior as model transitions that have been analyzed and found "secure".

In terms of the "design assurance chain" shown in Figure 1, there must exist documentation for

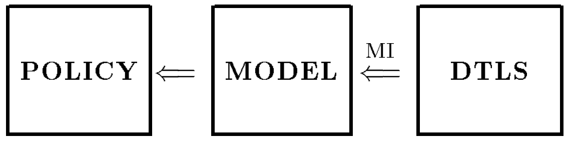

Figure 1: Design Assurance Chain

POLICY, MODEL, and DTLS, there must be arguments that the correspondences indicated by the arrows between them are available, and the ar-

*Work performed under contracts MDA-904-C-6017 and MDA-904-C-9999.

guments for correspondence must be *convincing*. The focus of this paper is Model Interpretation (MI), shown as the right-most arrow of Figure 1. All five conditions (policy, model, DTLS, and two arrows) are termed the "B2 checks".

Product evaluation under the *TCSEC* thus requires (a) a generic (formal security) policy; (b) a (formal security policy) model of that policy, together with results that assure its soundness and freedom from error; (c) a (descriptive top-level) specification of the TCB interface; and (d) a correspondence between the DTLS and the model, the model "interpretation".

Satisfaction of the B2 checks for a proprietary computing system builds directly on that system's idiosyncratic characteristics at the boundary of its Trusted Computing Base (TCB). Thus while general modeling resources can be used, the step of matching the system to the modeling entities is by nature specific to the system.

The trend in trusted products towards compliance with industry standards — especially the Portable Operating System Interface (POSIX) and the Standard Query Language (SQL) — makes possible a revision of the traditional procedure for satisfying the B2 checks. Rather than matching every system design directly to a model, one can produce a generic model interpretation for each industry standard, thereafter using the results to simplify and reduce the task of model interpretation for conforming implementations.

In terms of the generic "design assurance chain" in Figure 2, the Generic Model Interpretation

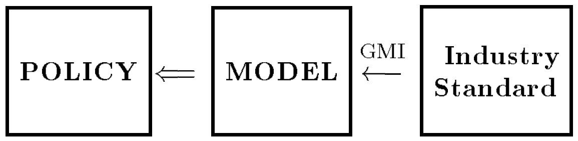

Figure 2: Generic Design Assurance Chain

(GMI) from the standard to the model will be available at the time that the product-specific model interpretation begins. Since a conforming implementation will match the specification of the standard by definition, a model interpretation task will be limited to any minor differences between the standard and the product. The production and review of the resulting model interpretation will be easier tasks, smaller tasks, and tasks that require substantially less time to perform.

This paper presents the results of a recent effort to create generic model interpretations for both POSIX and SQL (see [POSIX.1] and [LEFF91], respectively).

The next two sections describe the processes of producing and using a generic model interpretation. The following two sections describe the results of applying this process to POSIX.1 and SQL. The final section summarizes the work.

Producing a Generic Model Interpretation

The model interpretation portion of the B2 checks consists of matching part of the system's TCB interface to the security model being used. The results can be viewed conceptually as three tables. The first table (Figure 3) identifies which of the

Name	Interface?	Reason
module-001	no	...
module-002	yes	...
module-003	yes	...
⋮	⋮	⋮
module-998	no	...
module-999	yes	...

Figure 3: TCB Interface Table

TCB modules are visible at the TCB interface. The second table (Figure 4) identifies which of

Name	Model?	Reason
module-002	no	...
module-003	yes	...
module-017	no	...
⋮	⋮	⋮
module-952	no	...
module-999	yes	...

Figure 4: Modules to Model

the TCB-interface modules should be interpreted in modeling terms. The last table (Figure 5) lists the model rules (or transitions) that correspond to the TCB call.

The construction and justification of these three tables constitutes the bulk of the task of "model interpretation". In an ideal situation, the system or product would be stable and fully documented. There would also be resources available for answering questions and providing clarification. The

Name	Rule
module-003	ρ_3
module-042	ρ_{12}, ρ_5
module-116	ρ_4
⋮	⋮
module-666	ρ_{12}, ρ_7, ρ_{16}
module-999	ρ_7

Figure 5: Correspondence to Model

effort of constructing the first table would begin with a list of all the modules that constitute the TCB. Review of each of the modules would determine whether or not it is part of the TCB interface, and whether it is "visible" at the TCB interface. The subset of the modules identified as "at the interface" in the construction of the TCB-interface table would be the starting point for the modules-to-model table. The determination of whether a module should be modeled is more complex. It is dependent on such things as which system abstractions are exported by the TCB, which modules are available to ordinary users (rather than just to specially-privileged users), whether the module's action is part of access-control-policy mediation, and whether the module's action is a null action in a modeling context. Since the process of determining *whether* a module should be modeled is tightly coupled with the assignment of a corresponding model rule (or rules), the construction of the model-correspondence table will usually proceed in parallel with the construction of the modules-to-model table.

In less ideal situations (such as system model interpretations and product model interpretations while the system is still in flux), the same tables must be constructed, but the available materials are less complete. One must therefore proceed cautiously, realizing that many conclusions will have to be conditional and may have to be revised when more information becomes available.

Analysis of a generic industry standard entails a double-pronged effort. The first effort is the review of all the system calls according to their type. The set of types is developed inductively as the analysis proceeds. The second analytical approach is to assess the system calls with regard to their availability to different classes of users and to their relation to the enumerated policies that are of interest. This analysis is referred to as the "SAP" analysis.[1] A set of guidelines for SAP analysis is provided in the annex.

Consideration of both the types and SAP results will allow a determination of which TCB modules are the proper ones for generic model interpretation. The final step is identifying the correspondences between system calls and model rules to complete the construction of the generic model interpretation.

Using a Generic Model Interpretation

A generic model interpretation provides a starting point for the review of a model interpretation of a conforming implementation in the form of a Correspondence-to-Model table. A completed model interpretation for a specific implementation will also include a Correspondence-to-Model table. A comparison of those two tables will be the first step in evaluating the specific model interpretation.

Name	Rule
mod-n_1	ρ_{n_1}
mod-n_2	ρ_{n_2}
mod-n_3	ρ_{n_3}
⋮	⋮
mod-n_{k-1}	$\rho_{n_{k-1}}$
mod-n_k	ρ_k

GMI

Name	Rule
mod-m_1	ρ_{m_1}
mod-m_2	ρ_{m_2}
mod-m_3	ρ_{m_3}
⋮	⋮
mod-m_{k-1}	$\rho_{m_{k-1}}$
mod-m_k	ρ_{m_k}

MI

Figure 6: Comparison of Two Correspondences

The two model-correspondence tables have four modes of comparison:

1. an entry in the specific table matches an entry on the generic table exactly in both the "name" and "rules" value;

2. an entry in the specific table matches the "name" value, but the corresponding rules are not the same;

[1]The designation "SAP" derives from the codes used during the analysis: "S" refers to system calls that are available to a Standard user. "A" refers to system calls that include some extra effect for users with Appropriate privilege. "P" refers to system calls that relate to the enumerated Policies.

3. an entry in the specific table does not match any entry's "name" value in the generic table; and

4. an entry in the generic table does not match any entry's "name" value in the specific table.[2]

The generic model interpretation results apply to the specific implementation in the first case. In the other cases, the anomaly has to be resolved. The question is not "what mistake has been made in the specific model interpretation?" but "why is there this difference?" The reasons could be

- an error in generic model interpretation;
- an error in specific model interpretation; or
- differences between the specific case and the generic cases justify different results.

An initial comparison of the specific model interpretation table with the generic table has the beneficial effect of focusing attention on that portion of the modeling interpretation that is most in need of consideration.

The use of the generic interpretation is similar to a situation where one is attempting to construct a model interpretation during the completion of the implementation phase. In this case, the task is the construction of the model-correspondence table. The generic model-correspondence table provides version-zero of the required table. Meticulous review of the specific implementation in comparison to the generic model interpretation will allow the identification of both those rows that match the specific conforming implementation exactly and those system calls that need direct analysis and treatment.

GMI for POSIX.1

POSIX.1 is an especially lucrative target for generic model interpretation. Not only are there many trusted operating systems pledged to POSIX.1 compliance, but also there is voluminous open literature concerning Unix and POSIX.1 (see, for example, [POSIX.1], [USL92a], [USL92b]).

[2]There is the possibility, of course, that both the generic and the specific model interpretations are erroneous, but that their errors compensate, making the rows match each other. It is assumed that independent commission of such mistakes by model-interpretors is rare enough to be disregarded.

Types	**Code**
Access	A
A/policy control	A/C
A/post-mediation access	A/A
A/other factors	A/O
A/O/groups	A/O/
A/O/owner	A/O/O
A/O/pathname resolution	A/O/P
A/O/attributes	A/O/A
A/O/id's	A/O/I
Process	P
P/address space	P/A
P/memory	P/M
Job Control	J
J/signals	J/S
J/locks	J/L
J/pipes	J/P
Files	F
F/descriptors	F/D
F/link	F/L
IPC	I
I/msgs	I/M
I/sEmaphores	I/E
I/sHared memory	I/H
Resources	R
R/devices	R/D
R/D/STREAMS	R/D/S
R/affinity	R/A
R/file systems	R/F
Operations	O
O/networking	O/N
O/stat	O/S
O/auDit	O/D
O/auThentication	O/T
Magic	M
vfun's	V

Figure 7: "Types" of POSIX System Calls

System Call	**Corresponding Rules**
exit	release-access
accept	get-access
chmod	grant/rescind-access
close	release-access
connect	get-access
creat	create-object
lcntl	create/delete-object
mkmldir	create-objects
mknod	create-objects
mount	create-objects
sem_release_id	release-access, delete-object
setoacl	grant/rescind-access
setomac	change-object-level
dup	get-access
dup2	get-access
fchmod	grant/rescind-access
mkdir	create-object
mount	create-objects
msgctl	release-access, delete-object
msgget	create-object, get-access
open	get-access
rmdir	delete-object
semctl	release-access, delete-object
semget	create-object, get-access
shmat	get-access
shmctl	release-access, delete-object
shmdt	release-access
shmget	create-object, get-access
socket	create-object
socketpair	create-object
symlink	create-object
umount	delete-subtree
mknod	create-object

Figure 8: POSIX.1 Correspondence to Model

The situation of treating POSIX.1 generically is like the non-ideal situations above, but there is no expectation that more information will ever be available. That is, the details of a specific conforming implementation will never be available until that model interpretation is undertaken. Because of this intrinsic conditionality, the generic modeling interpretation will adopt a variation of the ideal approach described above.

In the context of a specific system, the full list of TCB modules will be a superset of the required POSIX.1 system calls. In the general case, therefore, the table of TCB modules constructed will necessarily be conditional and incomplete. The best that can be done is to construct a best-efforts modules-to-model table, a list of those POSIX.1 system calls that should usually be interpreted in modeling terms. From that conditional table, a corrresponding conditional model-correspondence table can be constructed.

In sum, the generic-model-interpretation plan for POSIX.1 is to construct a table of POSIX system calls that will normally be modeled, together with corresponding model rules.

Analysis of POSIX.1 identified the types shown in Figure 7.[3] The post-mediation access subtype was included with the mandatory types for model interpretation. The correspondence to model rules is shown in Figure 8.

GMI for SQL

SQL is also a lucrative target for model interpretation, but the wide variety in possibilities for Reference-Monitor-protected “objects” makes a comprehensive generic model interpretation a larger task than for POSIX.1. The results included in [MS96b] and summarized here constitute an initial step in generic model interpretation for SQL. The correspondences derived were produced by limiting the scope of attention to databases, tables, rows, view definitions, and columns.[4]

As is the case for POSIX.1, there is substantial open literature concerning SQL (see, for example, [BED93], [LEFF91]).

Analysis of SQL identified the types shown in Figure 9. The post-mediation access subtype was

[3]It is important to note that the types used for this analysis were defined subjectively and are in no way unique or necessary. Moreover, the types are not disjoint. In fact, some of the types were explicitly noted as overlapping with other types.

[4]Other object-candidates are indexes, constraints, and stored procedures.

Types	Code
Access	A
A/policy control	A/C
A/post-mediation access	A/A
A/other factors	A/O
SQL	SQL
Admin & Support	A&S
security	S
S/Grant,Revoke	S/GR
S/Audit Support	S/A
S/Tier 2 Audit Layer	S/A2
S/M.A.C.	S/M
S/D.A.C.	S/D
S/Discrete Privilege	S/P
Data Integrity	I
I/concurrency	I/C
I/transaction	I/T
I/recovery	I/R
I/mirroring	I/M
I/archiving	I/A
Object Management	O
O/DB Management	O/D
O/System Catalog	O/SC
O/Table	O/T
O/Row Data	O/R
O/Index	O/I
O/Constraint	O/C
O/View	O/V
O/Synonym	O/Y
O/Statistics	O/S
O/In-Core Dictionary	O/Lex
O/Stored Procedures	O/SP
Data	D
D/DB Services	D/D
D/B-tree	D/Bt
D/Row	D/R
Resource Management	R
R/DB space & chunk	R/D
R/Page	R/P
R/Page & slot	R/PS
R/Shared memory	R/SM
R/Buffer	R/B
R/Header	R/H
Magic	M
vfun's	V

Figure 9: "Types" of SQL Commands

SQL Call	**Corresponding Rules**
Change Process Label	change-object-level
Create Database	create-object
Drop Database	delete-object
Open-Lock Database	get-access
Close-Unlock Database	release-access
Create System Catalog	create-object
Open System Catalog Table	get-access
Drop System Catalog	delete-object
Read System Catalog	null transition
Write System Catalogs	null transition
Create Table	create-object
Drop Table	delete-object
Alter Table	null transition
Fast Alter Table	null transition
Open-Lock Table	get-access
Close Table	release-access
Insert Row	create-object
Delete Row	delete-object
Select Row	null transition
Update Row	null transition
Create View	create-object
Drop View	delete-object
Create Database Entry	create-object
Open Database Entry	get-access
Drop Database Entry	delete-object
Close Database	release-access
Grant Privilege	give-access
Revoke Privilege	rescind-access
Modify Database Label	change-object-level
Modify Table Sensitivity Label	change-object-level
Modify Row Sensitivity Label	change-object-level
Grant Table Level Privilege	give-access
Revoke Table Level Privilege	rescind-access
Grant Database Level Privilege	give-access
Grant Table Level Privilege	give-access
Revoke Database Level Privilege	rescind-access
Revoke Table Level Privilege	rescind-access

Figure 10: SQL Correspondence to Model

included with the mandatory types for model interpretation. The correspondence to model rules is shown in Figure 10.

Summary

The benefits of producing generic model interpretations as a tool to facilitate system and product evaluation are many. The work reported herein has both developed the theory of generic model interpretations and begun the task of providing generic model interpretations for POSIX.1 and SQL. The POSIX.1 results are very comprehensive, a result both of the more extensive information available on POSIX and of the narrower scope available for vendors producing a POSIX-compliant product that also meets B2 or above requirements from the *TCSEC*. The SQL results are also comprehensive for the object candidates addressed, but further work will be required to complete the consideration of all possible object candidates.

To the benefits provided by industry standards can now be added the ability to facilitate model interpretation preparation and review through the use of generic model interpretations.

References

[AND72] J. P. Anderson, “Computer Security Technology Planning Study”, ESD–TR–73–51, Vol. I, AD–758 206, ESD/AFSC, Hanscom AFB, MA, October 1972.

[BLP73] D. E. Bell and L. J. La Padula, “Secure Computer Systems: Mathematical Foundations”, MTR–2547, Vol. I, The MITRE Corporation, Bedford, MA, 1 March 1973. (ESD–TR–73–278–I)

[BELL73] D. E. Bell, “Secure Computer Systems: A Refinement of the Mathematical Model”, MTR–2547, Vol. III, The MITRE Corporation, Bedford, MA, December 1973. (ESD–TR–73–278–III)

[BLP75] D. E. Bell and L. J. La Padula, “Secure Computer Systems: Unified Exposition and Multics Interpretation”, MTR–2997, The MITRE Corporation, Bedford, MA, July 1975. (ESD–TR–75–306)

[BL86] D. E. Bell, ”Secure Computer Systems: A Network Interpretation”, Proc., 2nd Aerospace Conference, McLean, VA, 1986, 32-39.

[BL86] D. E. Bell, ”Trusted Xenix Interpretation: Phase 1”, Proc. 13th NCSC, Washington, DC, 1990, 333-339.

[BIBA77] K. Biba, “Integrity Considerations for Secure Computer Systems”, The MITRE Corporation, Bedford, MA, April 1977.

[BED93] J. S. Bowman, S. L. Emerson, M. Darnovsky, , *The Practical SQL Handbook*, 2nd ed. (Addison-Wesley: Reading, MA, 1993)

[GOME82] J. A. Goguen and J. Meseguer, “Security Policies and Security Models”, *Proc.* 1982 IEEE Symp. on Security and Privacy, Oakland, CA, April 26–28, 1982, 11–20.

[GOME84] J. A. Goguen and J. Meseguer, “Unwinding and Inference Control”, *Proc.* 1984 IEEE Symp. on Security and Privacy, Oakland, CA, April 29–May 2, 1984, 75–86.

[LPB73] L. J. La Padula and D. Elliott Bell, "Secure Computer Systems: A Mathematical Model", MTR–2547, Vol. II, The MITRE Corporation, Bedford, MA, 31 May 1973. (ESD–TR–73–278–II)

[LEFF91] *Using INFORMIX-SQL*, 2nd ed. (Addison-Wesley: Reading, MA, 1991)

[MS96a] "Generic Model Interpretation of POSIX.1", Mitretek Systems, McLean, VA, to appear.

[MS96b] "Generic Model Interpretation of SQL", Mitretek Systems, McLean, VA, to appear.

[POSIX.1] Information Technology – Portable Operating System Interface (POSIX) – Part 1: System Application Program Interface (API) [C Language]. ISO/IEC 9945-1:1990, September, 1990.

[TCSEC85] *Department of Defense Trusted Computer System Evaluation Criteria,* DoD 5200.28-STD, December 1985.

[TDI91] *Trusted Database Management Interpretation of the Trusted Computer System Evaluation Criteria,* NCSC-TG-021, Version 1, April 1991.

[USL92a] Programming with Unix System Calls: UNIX SVR4.2. (Prentice-Hall: Englewood Cliffs, NJ, 1992)

[USL92b] Operating System API Reference UNIX SVR4.2. (Prentice-Hall: Englewood Cliffs, NJ, 1992)

[WALT74] K. G. Walter, *et al.*"Primitive Models for Computer Security", ESD-TR-74-1147, Electronic Systems Division (MCIT), Air Force Systems Command, Hanscom AFM, Bedford, MA, January, 1974.

[WEIS69] C. Weissman, "Security Controls in the ADEPT-50 Time-Sharing System", *AFIPS Conf. Proc.* **35** FJCC 1969, 27–38.

SAP Analysis Guidelines

1. One can determine for each system security call whether a standard user (that is, one running without "appropriate privilege") can exercise the call and whether an administrative user (that is, one running with "appropriate privilege") can exercise the call with additional options or different effect.

2. The security policies that will be addressed in the model and modeling interpretation can be specified precisely.

3. For this exercise, the specified security policies are simple-security, discretionary-security, and information flow in the exact form of the *-property.

4. One can then determine whether each system security call relates to the enforcement of the specified policy (really, "policies").

5. System Calls that a standard user can invoke (indicated by S) and that relate to enforcement of the specified policies (indicated by P) will definitely be modeled.

6. System Security Calls that an administrative user can invoke (indicated by A) and that relate to enforcement of the specified policies (P) are options for modeling.

7. The decision to model an AP system security call will be addressed on a case-by-case basis. Back-of-the-envelope rationale for an inclusion or exclusion will be generated in the consideration of each AP system security call.

8. Calls that do not relate to the enforcement of the specified security policies and calls whose effects are invisible in the model's state will not be modeled.